CS 3100 – Models of Computation – Fall 2011
Assignment 10 Solutions

1. **(15 points)** Applying the Schröder-Bernstein (SB), show that the number of points in a 3-dimensional grid over Nat,

$$3dGrid = \{\langle x, y, z\rangle \mid x, y, z \in Nat\}$$

is the same as those in a 2-dimensional grid over Nat,

$$2dGrid = \{\langle x, y\rangle \mid x, y \in Nat\}.$$

Here of course, $Nat = \{0, 1, 2, \ldots\}$.

These being infinite sets, if we can find a correspondence (1-1, total, and onto) mapping from one set to the other, they would have the same cardinality.

Finding the above correspondence is made easy by the S-B theorem, requiring us to find only 1-1, total, and into maps both ways. Here are those maps:

- $3dGrid \rightarrow 2dGrid$: $\lambda(x, y, z) : \langle 2^x \times 3^y \times 5^z, 0\rangle$ which maps the triples (x, y, z) uniquely onto the x axis, keeping the y coordinate always 0. This is a total, 1-1, and 'into' map. It is total because it is defined for all (x, y, z). It is 1-1 because it never collapses two distinct (x, y, z) into the same image. It is 'into' because it never hits all the range points.

  itemize
- $2dGrid \rightarrow 3dGrid$: $\lambda(x, y) : \langle x, y, 0\rangle$, which simply puts 0 in the third coordinate. This is also total, 1-1, and into.

2. **(20 points)** Using the SB-theorem, present a way to count regular expressions over the alphabet $\{a, b\}$, expressing your answer as a cardinal number.

This is an infinite set. Let us find a correspondence to Nat, thus showing that Reg, the set of regular expressions over $\{a, b\}$ has cardinality $\aleph_0$.

- $Reg \rightarrow Nat$: Encode the ASCII characters that make up the regular expression string into a Nat by concatenating the ASCII codes of each symbol. Example: `(a+b)*` becomes the number in hex: 289743982942, by consulting a standard ASCII table.
- $Nat \rightarrow Reg$: For each natural number in Binary format, generate the RE under a homomorphism `lambda x: b if x==1 else a`.

3. **(20 points)** What is the cardinality of A_{CFG}? Prove your result using the SB theorem. Hint: find a way to map $\langle G, w\rangle$ into Nat; then find a way to map Nat into $\langle G, w\rangle$ pairs using a numeric-order enumeration.

$$A_{CFG} = \{\langle G, w\rangle \mid G \textit{ is a } CFG \textit{ and } w \textit{ is a string in the language of } G\}$$

- Again, map each $\langle G, w\rangle$ pair into a natural number by taking the string represented by G (written out in some standard format) concatenated with the string represented by w. One caveat: we don't ever want $\langle G_1, w_1\rangle$ and $\langle G_2, w_2\rangle$ to map to the same Nat by having G_1 be a prefix of G_2, which can allow $\langle G_1, w_1\rangle$ and $\langle G_2, w_2\rangle$ to read the same. This can be avoided by first converting G into a Nat, say x, then w into another Nat, say y, and encoding them as $2^x \times 3^y$.

- For sending Nat into $\langle G, w \rangle$ pairs, simply pick an arbitrary grammar G as the first component. The string w can then be generated according to `nthnumeric` from each Nat.

4. **(10 points)** Show that $INFINITE_{DFA}$ is a decidable language.

   To show that something is decidable, present an algorithm (pseudo-code plus English, or even entirely English is fine, because there will be no ambiguity). Ideally we must use a bulletted style to facilitate reading/grading. I often prefer this style made-up to look like a C function:

```
decider_inf_dfa(DFA D) {
 * If D is not syntactically correctly encoded, then REJECT
 * Build the finite-state machine (DFA) graph of D
 * Check whether D has a reachable loop which there is a reachable final state
   (the final state may be within the loop, or the loop may be en-route the final state)
   If so, ACCEPT
   else REJECT
}
```

   Now, `decider_inf_dfa` describes a TM that serves as the desired decider.

5. **(10 points)** Show that $NOODD_{DFA}$ is decidable.

```
decider_noodd_dfa(DFA D) {
 * If D is not syntactically correctly encoded, then REJECT
 * Generate D', a DFA that accepts all odd-length strings over the given alphabet
 * Intersect D with D', and ACCEPT if the intersection is empty; REJECT otherwise.
}
```

   Now, `decider_nood_dfa` describes a TM that serves as the desired decider.

6. **(10 points)** Show that A_{CFG} is decidable.

   Employ any parsing algorithm as the decider.

7. **(10 points)** Describe the working of an enumerator TM for NEQ_{CFG} in bulletted steps.

   - Given two CFGs G_1 and G_2, enumerate all strings over Σ^* in numeric order
   - Employ G_1 and G_2 to parse each string
   - List the $\langle G_1, G_2 \rangle$ pair whenever the parsing results on some string differs.
   - This is an enumerator TM for NEQ_{CFG}.